Desenvolvimento de um Sistema Distribuído com Comunicação Cliente-Servidor usando Sockets e RPC/gRPC

Simulação sistema de estoque de hortifruti

Brenda Martinez

Análise e Desenvolvimento de Sistemas

dezembro/2024

# Overview

* Essa documentação foi produzida utilizando o sistema operacional **WINDOWS**.
* Linguagem: Python 3
* Bibliotecas: socket, grpc, concurrent
* Instalando uma biblioteca python (Windows):
* Necessário ter o python previamente instalado
* Abrir o CMD e executar o comando “pip install {nome da biblioteca}”  
    
  Texto

  Descrição gerada automaticamente

# Como Executar o Sistema (Windows)

#### Implementação utilizando Sockets:

1. Abra o CMD do Windows e entre na pasta onde o arquivo do servidor está localizado.
2. Execute o servidor utilizando o comando “python servidor.py”  
     
   Texto

   Descrição gerada automaticamente
3. Abra uma nova guia no CMD do Windows e entre na pasta onde o arquivo do cliente está localizado.  
     
   Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

   Descrição gerada automaticamente
4. Execute o cliente utilizando o comando “python cliente.py”  
     
   Texto

   Descrição gerada automaticamente
5. Note que assim que o cliente é executado, o servidor retorna uma mensagem contendo o IP e a porta em que a conexão foi estabelecida.  
     
   Texto

   Descrição gerada automaticamente
6. Para interagir com o servidor, digite o comando desejado e aperte “Enter”. Para encerrar a conexão, digite “5” e aperte “Enter”.  
     
   Texto

   Descrição gerada automaticamente
7. Após 3 conexões, o servidor é fechado automaticamente.  
     
   Texto

   Descrição gerada automaticamente

FLUXOGRAMA

Diagrama

Descrição gerada automaticamente

#### Implementação com gRPC:

1. Abra o CMD do Windows e entre na pasta onde o arquivo “produtos.proto” está localizado.
2. Digite o comando “python -m grpc\_tools.protoc -I. --python\_out=. --grpc\_python\_out=. produtos.proto” para gerar os arquivos contendo o código gRPC (caso os mesmos já tenham sido gerados, ignorar esse passo, ou excluir os arquivos e executar o comando para gerar novamente).  
     
   Texto

   Descrição gerada automaticamente
3. Execute o servidor utilizando o comando “python servidor.py”  
     
   Interface gráfica do usuário, Texto

   Descrição gerada automaticamente  
     
   Ao executar o programa, ele irá gerar um arquivo de texto contendo a porta em que a conexão foi estabelecida.  
     
   Tela de celular

   Descrição gerada automaticamente
4. Abra uma nova guia no CMD do Windows e entre na pasta onde o arquivo do cliente está localizado (conforme ensinado anteriormente).
5. Execute o cliente utilizando o comando “python cliente.py”.  
     
   Texto

   Descrição gerada automaticamente
6. Para interagir com o servidor, digite o comando desejado e aperte “Enter”. Para encerrar a conexão, digite “5” e aperte “Enter”.  
     
   Texto

   Descrição gerada automaticamente  
     
   O servidor continuará aberto para novas conexões utilizando a mesma porta, conforme registro no arquivo “server\_port.txt”.

FLUXOGRAMA

Diagrama

Descrição gerada automaticamente

# Comparação entre as abordagens Sockets/gRPC:

A abordagem em sockets e a abordagem em gRPC apresentam diferenças significativas em termos de implementação, manutenção e eficiência, tanto do ponto de vista do servidor quanto do cliente. Ao utilizar sockets, a comunicação ocorre de forma direta via protocolo TCP, onde o cliente envia mensagens em texto simples, e o servidor interpreta, processa e retorna as respostas. Essa abordagem oferece flexibilidade e baixo nível de abstração, permitindo ao desenvolvedor controlar completamente o protocolo de comunicação. Contudo, essa liberdade vem acompanhada de desafios, como a necessidade de gerenciar manualmente conexões e interpretar mensagens. Essas tarefas tornam o código mais propenso a erros e difícil de manter, especialmente em sistemas que precisam escalar ou que possuem múltiplos clientes simultâneos.

Por outro lado, a abordagem com gRPC, baseada em Remote Procedure Call, utiliza o formato Protocol Buffers para serializar e desserializar mensagens. Isso abstrai muitos detalhes da comunicação, permitindo que o desenvolvedor foque na lógica da aplicação. No lado do servidor, o gRPC facilita o gerenciamento de múltiplas conexões, garantindo eficiência e segurança com suporte nativo a TLS. Além disso, o gRPC permite definir interfaces de serviço de forma declarativa, garantindo consistência e simplificando a manutenção. Do ponto de vista do cliente, a comunicação com o servidor é simplificada, já que as chamadas ao servidor se assemelham a chamadas de métodos locais, graças ao uso de stubs gerados automaticamente pelo Protobuf.

Enquanto sockets são mais adequados para implementações simples, gRPC é mais indicado para sistemas robustos e escaláveis. A abordagem em sockets demanda mais trabalho manual para implementar funcionalidades básicas, como a validação de mensagens e o suporte a múltiplos clientes, o que pode se tornar um obstáculo em projetos maiores. Por outro lado, o gRPC introduz dependências adicionais e requer aprendizado de ferramentas como Protobuf, mas compensa com uma interface clara, suporte nativo a múltiplas linguagens e um formato de mensagens mais eficiente.